POLSYS_GLP: A Parallel General Linear Product Homotopy Code for Solving Polynomial Systems of Equations

HAI-JUN SU, J. MICHAEL MCCARTHY
University of California, Irvine
MASHA SOSONKINA
Ames Laboratory
and
LAYNE T. WATSON
Virginia Polytechnic Institute and State University

Globally convergent, probability-one homotopy methods have proven to be very effective for finding all the isolated solutions to polynomial systems of equations. After many years of development, homotopy path trackers based on probability-one homotopy methods are reliable and fast. Now, theoretical advances reducing the number of homotopy paths that must be tracked, and in the handling of singular solutions, have made probability-one homotopy methods even more practical. POLSYS_GLP consists of Fortran 95 modules for finding all isolated solutions of a complex coefficient polynomial system of equations. The package is intended to be used on a distributed memory multiprocessor in conjunction with HOMPACK90 (Algorithm 777), and makes extensive use of Fortran 95 derived data types and MPI to support a general linear product (GLP) polynomial system structure. GLP structure is intermediate between the partitioned linear product structure used by POLSYS_PLP (Algorithm 801) and the BKK-based structure used by PHCPACK. The code requires a GLP structure as input, and although finding the optimal GLP structure is a difficult combinatorial problem, generally physical or engineering intuition about a problem yields a very good GLP structure. POLSYS_GLP employs a sophisticated power series end game for handling singular solutions, and provides support for problem definition both at a high level and via hand-crafted code. Different GLP structures and their corresponding Bezout numbers can be systematically explored before committing to root finding.

This work was supported in part by the Minnesota Supercomputing Institute, Air Force Office of Scientific Research grant F49620-02-1-0090, National Science Foundation grant DMI-0218285, and Air Force Research Laboratory grant F30602-01-2-0572.
Authors' addresses: H.-J. Su, J. M. McCarthy, Robotics and Automation Laboratory, University of California, Irvine, Irvine, CA 92697, suh@eng.uci.edu; M. Sosonkina, Ames Laboratory, Iowa State University, Ames, IA 50011, masha@scl.ameslab.gov; L. T. Watson, Departments of Computer Science and Mathematics, Virginia Polytechnic Institute & State University, Blacksburg, VA 24061-0106, ltw@cs.vt.edu.

Categories and Subject Descriptors: G.1.5 [**Numerical Analysis**]: Roots of Nonlinear Equations — *continuation* (*homotopy*) *methods, polynomials, systems of equations*; G.4 [**Mathematics of Computing**]: Mathematical Software

General Terms: Algorithms

Additional Key Words and Phrases: Chow-Yorke algorithm, curve tracking, fixed point, Fortran 95, general linear product, globally convergent, homotopy methods, linear product decomposition, probability-one, zero

1. INTRODUCTION

Polynomial systems of equations arise in many applications: robotics, computer vision, kinematics, chemical kinetics, truss design, geometric modeling, and many others (see [Morgan 1987] and [Verschelde 1996]). In applications where all the solutions, or a significant number of solutions, must be found, or when locally convergent methods fail, globally convergent, probability-one homotopy methods are preferred. Homotopy methods for polynomial systems were first proposed by Garcia and Zangwill [1977] and Drexler [1979]. While the method in [Garcia and Zangwill 1977] was easily demonstrated by topological techniques and the start system was easily solved, the homotopy produced many more paths than the total degree of the system. Drexler used the powerful results of algebraic geometry to prove his method. Two years later, using differential geometry, Chow, Mallet-Paret, and Yorke [1979] improved on the results of Garcia and Zangwill with a general homotopy which produced the same number of paths as the number of solutions (provided there are a finite number of them), counting multiplicities and solutions at infinity. The start system of this homotopy was difficult to solve. Morgan [1983] solved this problem with a much simpler start system, which had trivially obtained roots, and could be used in a general homotopy. The suggestion by Wright [1985] and Morgan [1986a], [1986b] to track the homotopy zero curves in complex projective space, rather than in Euclidean space, was another fundamental breakthrough—in complex projective space certain paths would no longer diverge to infinity (have infinite arc length), and paths in general were made shorter. Other notable publications, which appear around the end of the first decade of research, are by Meintjes and Morgan [1985], Tsai and Morgan [1985], and Watson, Billups, and Morgan [1987].

In roughly the past decade, since the development of robust, efficient homotopy path tracking algorithms, work has shifted towards lowering the number of paths that must be tracked. In essence, all the methods try to construct a start system for the homotopy map that better models the structure of the given polynomial system, the target system for the homotopy map. Early work includes m-homogeneous theory by Morgan and Sommese [1987a]. In m-homogeneous theory the powerful connection of probability-one homotopy methods for polynomials with the field of algebraic geometry is reestablished (see [Drexler 1979]) with the generalization of the classical theorem of Bezout. Generalizations of m-homogeneous theory appeared in [Verschelde and Haegemans 1993] with the GBQ method, and

in [Verschelde and Cools 1993] with set-structure analysis. The methods in both [Verschelde and Haegemans 1993] and [Verschelde and Cools 1993] are derived by modifying slightly the main theorem in [Morgan and Sommese 1987a], but are nonetheless important. The most recent definitive theoretical work is that of Morgan, Sommese, and Wampler [1995]. Though the theorems of [Morgan, Sommese, and Wampler 1995] are very powerful, used in their full generality, they suggest more an approach for exploiting structure than an algorithm. The method used here in POLSYS_GLP for constructing the start system is essentially the same as that in POLSYS_PLP [Wise, Sommese, and Watson 2000], which was based on the results of [Verschelde and Haegemans 1993] and [Morgan, Sommese, and Wampler 1995].

Attention has also been paid to the problem of calculating singular solutions of polynomial systems using homotopy methods. Approaches have been proposed based on Newton's method (see for example [Griewank 1985]), and based on complex analysis as in the work of Morgan, Sommese, and Wampler [1991], [1992a], [1992b]. The most useful approach of those mentioned is found in [Morgan, Sommese, and Wampler 1992b], where the foundation of a reasonable end game is laid. Other work has come from Sosonkina, Stewart, and Watson [1996]. Their approach, which is used in the polynomial system routine POLSYS1H of HOMPACK90 [Watson et al. 1997], is moderately successful on low, odd order singularities. To accurately compute a singular solution of order 30, say, requires a very sophisticated end game like that in [Morgan, Sommese, and Wampler 1992b], which POLSYS_GLP incorporates.

Publically available codes for solving polynomial systems of equations using globally convergent, probability-one homotopy methods do exist: HOMPACK [Watson, Billups, and Morgan 1987], written in FORTRAN 77, and HOMPACK90 [Watson et al. 1997], written in Fortran 90, both have polynomial system solvers. CONSOL in the book by Morgan [1987] is also written in FORTRAN 77. However, neither HOMPACK90 nor CONSOL has a sophisticated start system that can lower the number of homotopy paths that must be tracked below the total degree. The package PHCPACK by Verschelde [1997], written in Ada, allows a great variety of choices for the start system, and uses an end game like the one proposed in [Morgan, Sommese, and Wampler 1992b]. PHCPACK, based on BKK theory, has a distinctly combinatorial flavor, and tends to be rather slow on large scale production problems.

Polynomial structure is a complicated subject, attacked variously by the combinatorial BKK theory [Verschelde and Cools 1993] and algebraic geometry [Morgan, Sommese, and Wampler 1995]. A design choice of POLSYS_PLP [Wise, Sommese, and Watson 2000] was to strike a balance between the most general structural descriptions (yielding minimal numbers of paths to track, but extremely difficult algorithmically) and no structure at all (where the total degree number of paths must be tracked, algorithmically trivial). The trade-off is moot, because a search for structure may very well cost more than simply tracking the paths a fancier structure would have eliminated. Further, for many industrial problems, an m-homogeneous

structure or the more general partitioned linear product (PLP) structure supported by POLSYS_PLP is perfectly adequate, and often even optimal. The structure supported by POLSYS_GLP is called *general linear product*, which in generality lies between partitioned linear product and the arbitrary set-structure supported by PHCPACK. The motivation for this work and POLSYS_GLP is a significant class of problems [Su, McCarthy, and Watson 2004] for which the PLP structure is apparently inadequate.

An excellent source on homotopy methods in general is the book by Allgower and Georg [1990], and [Blum et al. 1998] contains many references oriented towards the complexity aspects of polynomial root finding.

2. POLYNOMIAL SYSTEMS OF EQUATIONS

Let $F(z) = 0$ be a polynomial system of n equations in n unknowns. In symbols,

$$F_i(z) = \sum_{j=1}^{n_i} \left[c_{ij} \prod_{k=1}^{n} {z_k}^{d_{ijk}} \right] = 0, \qquad i = 1, \ldots, n, \tag{1}$$

where the c_{ij} are complex (and usually assumed to be different from zero) and the d_{ijk} are nonnegative integers. The *degree* of $F_i(z)$ is

$$d_i = \max_{1 \le j \le n_i} \sum_{k=1}^{n} d_{ijk},$$

and the *total degree* of the system (1) is

$$d = \prod_{i=1}^{n} d_i.$$

Define $F'(w)$ to be the homogenization of $F(z)$:

$$F'_i(w) = {w_{n+1}}^{d_i} \, F_i(w_1/w_{n+1}, \ldots, w_n/w_{n+1}), \qquad i = 1, \ldots, n. \tag{2}$$

Note that, if $F'(w^0) = 0$, then $F'(\alpha w^0) = 0$ for any complex scalar α. Therefore, "solutions" of $F'(w) = 0$ are (complex) lines through the origin in $\mathbf{C}^{n+1}$. The set of all lines through the origin in $\mathbf{C}^{n+1}$ is called complex projective n-space, denoted $\mathbf{P}^n$, and is a compact n-dimensional complex manifold. (Note that we are using complex dimension: $\mathbf{P}^n$ is $2n$-dimensional as a real manifold.) The solutions of $F'(w) = 0$ in $\mathbf{P}^n$ are identified with the solutions and solutions at infinity of $F(z) = 0$ as follows: If $L \in \mathbf{P}^n$ is a solution to $F'(w) = 0$ with $w = (w_1, w_2, \ldots, w_{n+1}) \in L$ and $w_{n+1} \neq 0$, then $z = (w_1/w_{n+1}, w_2/w_{n+1}, \ldots, w_n/w_{n+1}) \in \mathbf{C}^n$ is a solution to $F(z) = 0$. On the other hand, if $z \in \mathbf{C}^n$ is a solution to $F(z) = 0$, then the line through $w = (z, 1)$ is a solution to $F'(w) = 0$ with $w_{n+1} = 1 \neq 0$. The standard definition of *solutions to* $F(z) = 0$ *at infinity* is simply *solutions to* $F'(w) = 0$ (*in* $\mathbf{P}^n$) *generated by* w *with* $w_{n+1} = 0$.

A solution $\hat{w} \in \mathbf{P}^n$ is called *geometrically isolated* if there exists an open ball $B \subset \mathbf{P}^n$, with $\hat{w} \in B$ and no other solutions in B. If no such ball exists, then the solution $\hat{w}$ is said to exist on a *positive dimensional solution set.* Suppose $\hat{w}$ is a geometrically

isolated solution to (2), and suppose B is a ball that contains it. For almost all perturbations of the coefficients of the polynomial, the perturbed polynomial has only nonsingular solutions. For all such sufficiently small perturbations of the coefficients, there exists a finite number m of solutions inside B to the perturbed system of equations. This number m is the *multiplicity* of the solution $\hat{w}$ to (2). A solution $\hat{z} \in \mathbf{C}^n$ to (1) is *singular* if the Jacobian matrix at $\hat{z}$, $D_z F(\hat{z})$, is singular, and *nonsingular* otherwise. Singular solutions at infinity are defined analogously in terms of coordinate patches [Morgan 1987]. A solution has multiplicity greater than one precisely when it is singular [Morgan 1987].

Now that the solution set has been described, the following beautiful result can be stated [van der Waerden 1953]:

BEZOUT'S THEOREM. *There are no more than d isolated solutions to $F'(w)$ in $\mathbf{P}^n$. If $F'(w) = 0$ has only a finite number of solutions in $\mathbf{P}^n$, it has exactly d solutions, counting multiplicity.*

In practical problems, finite, nonsingular, geometrically isolated solutions are of great importance and interest; they are also the easiest to deal with in the homotopy setting. However, since it is not possible *a priori* to separate the nonsingular solutions from the singular solutions and solutions at infinity, homotopy algorithms are forced to deal with the latter. Solutions that are singular or at infinity can cause serious numerical difficulties and inefficiency—these two types of solutions are discussed in later sections. More importantly, the problem of handling these "bad" solutions pales in comparison to the potentially huge number of solutions (and homotopy zero curves that must be tracked). The total degree d, called the Bezout number or more precisely the 1-homogeneous Bezout number [Morgan and Sommese 1987a], can be overwhelming even for tame-looking problems. For example, 20 cubic equations would have $d = 3^{20} \approx 3.5 \times 10^9$. Consequently, recent research has looked for methods that shrink (in a rigorous sense) the number of solutions that must be computed, while still retaining all the finite isolated solutions. *Reduction*, which seeks to lower the dimension of the system, is one approach that will work, but is not discussed here (see Chapter 7 of [Morgan 1987]). Sophisticated mathematical approaches, generally speaking, seek to "factor out" a significant portion of the nonphysical solutions (typically, including many solutions at infinity and multiplicities). For many important practical problems this is possible, since often systems that arise from physical models have symmetries and redundancies (which spawn solutions at infinity and multiple solutions), yet only a small number (compared to d) of finite, nonsingular solutions [Morgan and Sommese 1987b], [Verschelde 1996], [Verschelde and Cools 1993]. The next section discusses one such approach to reducing the number of homotopy zero curves that must be tracked: the general linear product (GLP) homotopy.

3. HOMOTOPIES FOR POLYNOMIAL SYSTEMS

Define a homotopy map $\rho : [0, 1) \times \mathbf{C}^n \to \mathbf{C}^n$ by

$$\rho(\lambda, z) = (1 - \lambda)G(z) + \lambda\, F(z). \tag{3}$$

$\lambda \in [0, 1)$ is the *homotopy parameter*, $G(z) = 0$ is the *start system*, and $F(z) = 0$ is the *target system*. The goal is to find a start system with the same structure as the target system, while possessing the property that $G(z) = 0$ is easily solved. In this section a start system with a *general linear product* (GLP) structure will be constructed.

Let $P = (P_1, P_2, \ldots, P_n)$ be an n-tuple of (topological) coverings P_i of the set $\{z_1, z_2, \ldots, z_n\}$. That is, for $i = 1, 2, \ldots, n$, $P_i = \{S_{i1}, S_{i2}, \ldots, S_{im_i}\}$, where S_{ij} has cardinality $n_{ij} \neq 0$ and $\bigcup_{j=1}^{m_i} S_{ij} = \{z_1, z_2, \ldots, z_n\}$. For clarity, P is called the *system covering*, and the P_i are the *component coverings*. For $i = 1, \ldots, n$, assume that the component F_i has the representation

$$F_i = \sum_{k=1}^{r_i} \prod_{j=1}^{m_i} p_{ijk},$$

where each polynomial p_{ijk} only involves variables from the set S_{ij}. For $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, m_i$ define d_{ij} to be the maximum degree of the polynomials $p_{ij1}, \ldots, p_{ijr_i}$. Thus if $F_2(z_1, z_2, z_3) = (3z_1 - z_2)(7z_2 - z_3) + (z_1 + z_2)(z_2^2 - z_2 z_3 + z_3)$, $S_{21} = \{z_1, z_2\}$, and $S_{22} = \{z_2, z_3\}$, then $d_{21} = 1$, $d_{22} = 2$. It is convenient, though only for the definition of the start system, to rename the variables component-by-component. Let $S_{ij} = \{z_{ij1}, z_{ij2}, \ldots, z_{ijn_{ij}}\}$. With all this said, the start system is represented mathematically by $G_i(z) = \prod_{j=1}^{m_i} G_{ij}$, where

$$G_{ij} = \begin{cases} \left(\sum_{k=1}^{n_{ij}} c_{ijk} z_{ijk} \right)^{d_{ij}} - 1, & \text{if } d_{ij} > 0; \\ 1, & \text{if } d_{ij} = 0, \end{cases} \qquad i = 1, 2, \ldots, n, \tag{4}$$

where the numbers $c_{ijk} \in \mathbf{C}_0 = \mathbf{C} \backslash \{0\}$ are chosen at random. The structure defined by the system covering P and manifested in (4) is called the *general linear product* structure. The degree of $G_i(z)$ is

$$\deg(G_i) = \sum_{j=1}^{m_i} d_{ij}.$$

Note that $d_i \leq \deg(G_i)$ always holds—this fact will be important later, when the projective transformation of the homotopy map is defined.

This start system is modeled after the one in [Wampler 1994], and, like its model, is desirable because it is computationally efficient and its solutions, all of which are obtained by the solution of a complex linear system, are nonsingular. To be precise, the linear subsystems into which $G(z) = 0$ decomposes, whether solvable or unsolvable, can be uniquely characterized by two lexicographic vectors. The first, $\Phi = (\Phi_1, \Phi_2, \ldots, \Phi_n)$, is called the *factor lexicographic vector*, and the second, $\Delta = (\Delta_1, \Delta_2, \ldots, \Delta_n)$, is called the *degree lexicographic vector*, where $(1, 1, \ldots, 1) \leq \Phi \leq (m_1, m_2, \ldots, m_n)$, and where, given Φ and all $d_{j\Phi_j} \neq 0$, $(0, 0, \ldots, 0) \leq \Delta \leq (d_{1\Phi_1} - 1, d_{2\Phi_2} - 1, \ldots, d_{n\Phi_n} - 1)$. For example, suppose that the

lexicographic pair (Φ, Δ) with all $d_{j\Phi_j} \neq 0$ is given. Then the linear system this pair uniquely represents is

$$A_\Phi z = \begin{pmatrix} \sum_{k=1}^{n_{1\Phi_1}} c_{1\Phi_1 k} z_{1\Phi_1 k} \\ \sum_{k=1}^{n_{2\Phi_2}} c_{2\Phi_2 k} z_{2\Phi_2 k} \\ \vdots \\ \sum_{k=1}^{n_{n\Phi_n}} c_{n\Phi_n k} z_{n\Phi_n k} \end{pmatrix} = \begin{pmatrix} e^{2\pi i(\Delta_1/d_{1\Phi_1})} \\ e^{2\pi i(\Delta_2/d_{2\Phi_2})} \\ \vdots \\ e^{2\pi i(\Delta_n/d_{n\Phi_n})} \end{pmatrix} \equiv b_\Delta, \tag{5}$$

where the z_{ijk} are as defined above. Either A_Φ is generically nonsingular, that is, nonsingular for almost all choices of the c_{ijk} from $\mathbf{C}_0$, or structurally singular. If A_Φ is structurally singular, then it contributes no solutions to $G(z) = 0$ and may be ignored. If A_Φ is generically nonsingular, then $A_\Phi z = b_\Delta$ has a unique solution for each Δ such that $(0, 0, \ldots, 0) \leq \Delta \leq (d_{1\Phi_1} - 1, d_{2\Phi_2} - 1, \ldots, d_{n\Phi_n} - 1)$. If some $d_{j\Phi_j} = 0$, then the factor $G_{j\Phi_j} = 1$ cannot be zero and A_Φ need not even be considered.

In order to count the number of solutions of $G(z) = 0$, it must be determined for each Φ whether or not A_Φ is generically nonsingular. If A_Φ is nonsingular then $\prod_{i=1}^{n} d_{i\Phi_i}$ is added to the "root count." The final root count is the total number of solutions B_{GLP} to $G(z) = 0$ and is called the GLP Bezout number. There is a combinatorial formula for determining whether or not A_Φ is generically invertible [Verschelde and Cools 1993]. For large problems, however, this rule is expensive. Computational experience with the approach of POLSYS_PLP, which uses numerical linear algebra with random real matrices to determine the generic invertibility of A_Φ, has proved that algorithm to be both efficient and reliable. Therefore POLSYS_GLP uses the same algorithm as POLSYS_PLP to determine the generic invertibility ofA_Φ. The issues of how to choose the c_{ijk} for such a method, and the likelihood of a generically invertible A_Φ being numerically singular, are discussed in detail in Wise, Sommese, and Watson [2000].

The importance of the number B_{GLP} derives from the theory explained at length in Wise, Sommese, and Watson [2000]. For clarity and completeness, some of the theorems, definitions, and discussion from [Wise, Sommese, and Watson 2000] will be recapitulated here.

THEOREM 3.1 [MORGAN, SOMMESE, AND WAMPLER 1995]. *Let $f : \mathbf{C}^n \to \mathbf{C}^n$ be a system of polynomials and $U \subset \mathbf{C}^n$ be (Zariski) open. Define $N(f, U)$ to be the number of nonsingular solutions to $f = 0$ that are in U. Assume that there are positive integers $r_1, \ldots, r_n$ and $m_1, \ldots, m_n$ and finitely generated complex vector spaces V_{ij} of polynomials for $i = 1, \ldots, n$ and $j = 1, \ldots, m_i$, such that*

$$f_i = \sum_{k=1}^{r_i} \prod_{j=1}^{m_i} p_{ijk}, \tag{6}$$

where $p_{ijk} \in V_{ij}$ *for* $i = 1, \ldots, n$, $j = 1, \ldots, m_i$ *and* $k = 1, \ldots, r_i$. *Let a system* g *be defined by* $g_i = \prod_{j=1}^{m_i} g_{ij}$, *with each* g_{ij} *a generic choice from* V_{ij}. *Then*

$$N(f, U) \leq N(g, U), \tag{7}$$

and (7) *is equality if, for each* i *with* $1 \leq i \leq n$, *there is a positive integer* k_i *such that the* $p_{ijk_i} \in V_{ij}$ *are generic for* $j = 1, \ldots, m_i$. *Also,* $g(z) = 0$ *is a suitable start system for the polynomial homotopy* $h(t, z) = (1 - t)f(z) + tg(z)$ *to find all nonsingular solutions to* $f(z) = 0$.

REMARK 3.1. The reader will have noted that the homotopy of Theorem 3.1 is different from the one given in (3), but this difference is only a cosmetic change of variables $t = 1 - \lambda$. Moreover, the last line of Theorem 3.1 will be made precise in Section 4.

REMARK 3.2. The subsystems of $g = 0$ are the systems $\hat{g} = (g_{1j_1}, \ldots, g_{nj_n}) = 0$ with $1 \leq j_i \leq m_i$ and $i = 1, \ldots, n$ (see Remark 1.1 from [Morgan, Sommese, and Wampler 1995]). In practice, one chooses g so that the subsystems $\hat{g} = 0$ are easy to solve, e.g., so that solving $\hat{g} = 0$ reduces to solving a linear system.

Let $\{e_{ijl} \mid l = 1, \ldots, \ell_{ij}\}$ denote a basis of the vector space V_{ij}. For $i = 1, \ldots, n$ and $j = 1, \ldots, m_i$ define

$$B_{ij} = \{z \mid e_{ijl}(z) = 0 \text{ for } l = 1, \ldots, \ell_{ij}\}.$$

Following [Morgan, Sommese, and Wampler 1995], the *bases have no pairwise intersection with* U if for any choice of j' and j'' with $1 \leq j' < j'' \leq m_i$,

$$B_{ij'} \cap B_{ij''} \cap U = \emptyset.$$

The next theorem relates the nonsingular solutions of $g(z) = 0$ with those of its subsystems.

THEOREM 3.2 [MORGAN, SOMMESE, AND WAMPLER 1995]. *Let* g *and* V_{ij} *be as in Theorem 3.1. Then*

1. $z_0 \in U$ *is a nonsingular solution to* $g(z) = 0$ *if and only if* z_0 *is a solution to exactly one subsystem of* $g(z) = 0$ *and it is a nonsingular solution to this subsystem.*
2. *Assume that the bases for the* V_{ij} *have no pairwise intersection with* U. *Then, if* $z_0 \in U$ *is a nonsingular solution to some subsystem of* $g(z) = 0$, *it is a solution to exactly one subsystem of* $g(z) = 0$.

REMARK 3.3. It follows from Theorem 3.2 that

$$N(g, U) \leq \sum_{\substack{1 \leq j_1 \leq m_1 \\ 1 \leq j_2 \leq m_2 \\ \vdots \\ 1 \leq j_n \leq m_n}} N\big((g_{1j_1}, g_{2j_2}, \ldots, g_{nj_n}), U\big), \tag{8}$$

with equality if the bases have no pairwise intersection with U. (This is Remark 2.2 from [Morgan, Sommese, and Wampler 1995].)

Suppose that P is a system covering as before. The vector spaces of polynomials V_{ij} will be constructed using P: consider both S_{ij} and d_{ij} for $j = 1, 2, \ldots, m_i$ and $i = 1, 2, \ldots, n$, and define V_{ij} to be the complex vector space of polynomials generated by the monomials in the variables from S_{ij} up to degree d_{ij} and the constant 1. For example, suppose that $f(z) = 0$ is a polynomial system in four variables for which $P_2 = \{S_{21}, S_{22}\}$, $S_{21} = \{z_2, z_3, z_1\}$, $S_{22} = \{z_3, z_4\}$, $d_{21} = 2$, and $d_{22} = 3$. Then

$$V_{21} = \mathbf{C}\langle z_2^2, z_3^2, z_1^2, z_2z_3, z_2z_1, z_3z_1, z_2, z_3, z_1, 1\rangle,$$
$$V_{22} = \mathbf{C}\langle z_3^3, z_4^3, z_3^2z_4, z_3z_4^2, z_3^2, z_4^2, z_3z_4, z_3, z_4, 1\rangle.$$

Choosing the V_{ij}, albeit implicitly, from P, with $U = \mathbf{C}^n$, ensures that the bases have no pairwise intersection with U. Since $G_{ij} \in V_{ij}$ is generic,

$$N(F, \mathbf{C}^n) \leq N(G, \mathbf{C}^n) = \sum_{\substack{1\leq j_1\leq m_1\\ 1\leq j_2\leq m_2\\ \vdots\\ 1\leq j_n\leq m_n}} N\big((G_{1j_1}, G_{2j_2}, \ldots, G_{nj_n}), \mathbf{C}^n\big) = B_{GLP}. \quad (9)$$

This entire discussion would be moot if B_{GLP} were not in many cases smaller than the total degree d. In fact, for many practical problems for well chosen V_{ij}, $B_{PLP} \ll d$, and for a class of important applications, also $B_{GLP} \ll B_{PLP} \ll d$. The computational implications for the homotopy map (3) with the start system $G(z) = 0$ are clear—only B_{GLP} homotopy zero curves must be tracked.

Since the start system of Theorem 3.1 can result in a lower number of paths to be tracked, while guaranteeing that paths will reach all nonsingular solutions of $f(z) = 0$, the number $N(g, \mathbf{C}^n)$ is commonly referred to as a generalized Bezout number. The GLP method explained here is but one way of arriving at such a number. The name *general linear product* is essentially a description of the structure of the start system $G(z) = 0$. There is a hierarchy of start system construction methods and corresponding generalized Bezout numbers $N(G, \mathbf{C}^n)$, based on start system complexity:

- 1-homogeneous,
- m-homogeneous,
- partitioned linear product,
- general linear product decomposition,
- general product decomposition.

The general linear product decomposition corresponds to each set P_i being a (topological) *covering* of the set $\{z_1, \ldots, z_n\}$. The special case where each P_i is a *partition* ($S_{ij_1} \cap S_{ij_2} = \emptyset$ for $j_1 \neq j_2$) is the partitioned linear product (PLP) decomposition. The m-homogeneous case is where all the P_i are the same partition, and 1-homogeneous is just m-homogeneous with $m = 1$. The method of greatest

generality is the general product decomposition (GPD). GPD is any method that utilizes Theorem 3.1 in more generality than GLP, so the start system does not reduce to a product of linear systems. GLP is exactly equivalent to the set-structure analysis of Verschelde and Cools [1993].

Finding the lowest possible GLP Bezout number is a challenging problem. There is no way, short of an exhaustive search through all possible system covers, of knowing which system cover P will give the lowest value of B_{GLP}. There are heuristics for picking the system covering P, but usually physical or engineering insight into the problem structure is a better guide. Even if B_{GLP} is not minimized, it may still be small enough so that the path tracking is computationally tractable.

4. THE PROBABILITY ONE ASPECT

Suppose that P is a system covering for (1) corresponding to the GLP Bezout number B_{GLP}. The following theorem demonstrates the probability-one aspect of the homotopy method in POLSYS_GLP.

THEOREM 4.1. *For almost all choices of c_{ijk} in the start system defined by (4), $\rho^{-1}(0)$ consists of B_{GLP} smooth curves emanating from $\{0\} \times \mathbf{C}^n$, which either diverge to infinity as λ approaches 1 or converge to solutions of $F(z) = 0$. Each nonsingular solution of $F(z) = 0$ will have a curve converging to it.*

Theorem 4.1 is essentially a restatement of the last line of Theorem 3.1, but deserves emphasis; its proof can be found in Section A.5 in the appendix of [Morgan, Sommese, and Wampler 1995]. A noteworthy observation is that since the homotopy map ρ is complex analytic, the homotopy parameter λ is monotonically increasing as a function of arc length along the homotopy zero curves starting at $\lambda = 0$ [Morgan 1987]. Thus, the homotopy zero curves never have turning points with respect to λ.

Though B_{GLP} may be much smaller than the total degree, the possibility of tracking paths of (3) that diverge to infinity still exists. These paths pose significant computational challenges, since time is wasted on divergent paths, and large magnitude solutions may not be found if a path is terminated prematurely. Tracking paths in complex projective space, which was originally proposed in [Morgan 1986a, 1986b], eliminates these concerns. With a suitable "projective transformation" no paths diverge to infinity as λ approaches 1. Moreover, though not guaranteed, paths tend be shorter in projective space.

Constructing the projective transformation is straightforward [Morgan 1986a, 1986b], [Watson, Billups, and Morgan 1987]. As with the homogenization of $F(z)$, define the homogenization of $\rho(\lambda, z)$ to be

$$\rho_i'(\lambda, w) = w_{n+1}^{\deg(G_i)} \rho_i \left(\lambda, \frac{w_1}{w_{n+1}}, \ldots, \frac{w_n}{w_{n+1}} \right), \qquad i = 1, \ldots, n.$$

Define the linear function

$$u(w_1, \ldots, w_{n+1}) = \xi_1 w_1 + \xi_2 w_2 + \ldots + \xi_{n+1} w_{n+1},$$

where the numbers $\xi_i \in \mathbf{C}_0$ are chosen at random. The *projective transformation* of $\rho(\lambda, z)$ is

$$\rho''(\lambda, w) = \begin{pmatrix} \rho'_1(\lambda, w) \\ \rho'_2(\lambda, w) \\ \vdots \\ \rho'_n(\lambda, w) \\ u(w) - 1 \end{pmatrix}.$$

That the projective transformation can be applied to the homotopy map ρ, without changing the essence of Theorem 4.1, follows from Remark 1.4 of [Morgan, Sommese, and Wampler 1995]. The precise statement follows.

THEOREM 4.2. *For almost all choices of the c_{ijk} in the start system defined by (4) and almost all choices of the ξ in the linear function $u(w)$, $(\rho'')^{-1}(0)$ consists of B_{GLP} smooth curves emanating from $\{0\} \times \mathbf{C}^{n+1}$, which converge to solutions of $F'(w) = 0$. Each nonsingular solution of $F'(w) = 0$ will have a curve converging to it.*

Henceforth, Theorem 4.2 will tacitly be the operative theorem, and references to ρ tacitly assume that the computer implementation actually works with ρ''.

5. HOMOTOPY PATH TRACKING AND THE END GAME

Theorem 4.1 says that in order to reach the nonsingular solutions of $F(z) = 0$, "smooth" (nonintersecting, nonbifurcating) paths in $\rho^{-1}(0)$ must be tracked. There are fast, reliable ways of doing this numerically. Three different path tracking algorithms (ordinary differential equation based, normal flow, and augmented Jacobian matrix) are described in [Watson, Billups, and Morgan 1987] and [Watson et al. 1997]. Simple linear-predictor, Newton-corrector methods are described in [Morgan 1987]and [Verschelde 1997]. There is compelling evidence favoring higher order methods and the normal flow algorithm over simpler schemes [Lundberg and Poore 1991], [Morgan, Sommese, and Watson 1989], [Watson, Billups, and Morgan 1987], [Watson et al. 1997]. As does POLSYS_PLP, POLSYS_GLP uses the sophisticated homotopy zero curve tracking routine `STEPNX` from HOMPACK90. The curve tracking details are identical to those of POLSYS_PLP [Wise, Sommese, and Watson 2000], and thus need not be repeated here.

The projective transformation eliminates diverging paths, but in doing so may give paths leading to highly singular solutions at infinity. When the curve being tracked converges to a multiple solution or a positive dimensional solution set of $F(z) = 0$ at $\lambda = 1$, necessarily rank $D\rho(\lambda, z) < n$, which affects both numerical stability and the rate of convergence of the corrector iteration (11). Newton-type algorithms may do very well with nonsingular solutions, but incur a significant expense at singular solutions, an order of magnitude worse than at nonsingular solutions. The end game (that phase of the algorithm when $\lambda \approx 1$) in POLSYS_GLP is based on theory in [Morgan, Sommese, and Wampler 1992b], and is identical to the end game in POLSYS_PLP. It provides reasonably accurate estimates of

a singular solution at a fairly low computational cost (compared to Newton-type algorithms). The supporting theory and complete algorithmic details are provided in the POLSYS_PLP paper [Wise, Sommese, and Watson 2000]. The end game for singular solutions in POLSYS_PLP represents a major advance over the algorithms available in POLSYS1H (HOMPACK90) and PHCPACK, and is a major contributor to the empirically observed efficiency of POLSYS_PLP.

6. PARALLEL IMPLEMENTATION

It is common for applications to have thousands or millions of paths to be tracked, and thus using parallel computation is natural. POLSYS_GLP is based on MPI-2, and is designed to run on massively parallel distributed memory multiprocessors. (If a problem is small enough that a serial version of POLSYS_GLP would be practical, then simply use the serial code POLSYS_PLP instead. Only for very large problems is the difference between B_{GLP} and B_{PLP} significant.) Although the homotopy paths are independent of each other, the path tracking times can vary enormously between paths, and the most efficient use of parallel resources is not immediately apparent. Some early work in parallel homotopy algorithms [Allison et al. 1989a], [Allison et al. 1989b], [Chakraborty et al. 1991], [Chakraborty et al. 1993], [Morgan and Watson 1989], [Pelz and Watson 1989] systematically compared shared memory and distributed memory implementations, and various coarse grained and fine grained ways of decomposing the path tracking computations. The conclusion from all that work, at least for the special case of polynomial systems, was that a message passing, master/slave paradigm with one complete path tracking task per slave was the best paradigm. Thus this is the parallel paradigm used by POLSYS_GLP.

The master computes all the start points, sending them one by one to idle slaves. The master also accumulates all the zeros and path information (error flags, arc length, number of function evaluations, projective coordinates) from the slaves. A slave waits for a start point, then tracks a complete path to $\lambda = 1$ or until an error condition occurs, then returns the zero and path information to the master, and waits for the next start point. Because of the variability in the path tracking times, the master is not a bottleneck even for hundreds of slave processors. Should the master become a bottleneck, the start point computation can be distributed, so a tree of masters could be used to spread the work of interacting with the slaves.

7. ORGANIZATION AND USAGE

The package POLSYS_GLP consists of two Fortran 95 modules (`GLOBAL_GLP`, `POLSYS2`—the latter name is chosen to avoid a conflict with the POLSYS_PLP module `POLSYS`). `GLOBAL_GLP` contains Fortran 95 derived data types to define the target system, the start system, and the system covering. As its name suggests, `GLOBAL_GLP` provides data globally to the routines in POLSYS_GLP. The module `POLSYS2` contains three subroutines: `POLSYS_GLP`, `BEZOUT_GLP`, and `SINGSYS_GLP`. `POLSYS_GLP` finds the root count (the Bezout number B_{GLP} for a given system covering P) and the roots of a polynomial system, `BEZOUT_GLP` finds only the root count.

SINGSYS_GLP checks the singularity of a given start subsystem, and is of interest only to expert users. The package uses the HOMPACK90 modules REAL_PRECISION, HOMPACK90_GLOBAL, and HOMOTOPY [Watson et al. 1997], the HOMPACK90 subroutine STEPNX, and numerous LAPACK and BLAS subroutines [Anderson et al. 1995]. The physical organization of POLSYS_GLP into files is described in a README file that comes with the distribution.

Arguments to POLSYS_GLP include an input tracking tolerance TRACKTOL, an input final solution error tolerance FINALTOL, an input singularity tolerance SINGTOL for the root counting algorithm, input parameters for curve tracking, various output solution statistics, and four Fortran 95 optional arguments: NUMRR, RECALL, NO_SCALING, and USER_F_DF. The integer NUMRR specifies the number of iterations times 1000 that the path tracker is allowed; the default value is 1. The logical variable RECALL should be included if, after the first call, POLSYS_GLP is being called again to retrack a selected set of curves. The presence of the logical variable NO_SCALING (regardless of value) causes POLSYS_GLP *not* to scale the target polynomial system. The logical optional argument USER_F_DF specifies that the user is supplying hand-crafted code for function and Jacobian matrix evaluation—this option is recommended if efficiency is a concern, or if the original formulation of the system is other than a linear combination of monomials.

POLSYS_GLP takes full advantage of Fortran 95 features. For example, all real and complex type declarations use the KIND specification; derived data types are used for storage flexibility and simplicity; array sections, automatic arrays, and allocatable arrays are fully utilized; interface blocks are used consistently; where appropriate, modules, rather than subroutine argument lists, are used for data association; low-level linear algebra is done with Fortran 95 syntax rather than with BLAS routines; internal subroutines are used extensively with most arguments available via host association. POLSYS_GLP is easy to use, with a short argument list, and the target system $F(z)$ defined with a simple tableau format (unless the optional argument USER_F_DF is present). The calling program requires the statement

```
USE POLSYS2
```

The typical use of POLSYS_GLP is either to call BEZOUT_GLP to obtain the root count B_{GLP} of a polynomial system of equations for a specified system covering P, or to call POLSYS_GLP to obtain all the roots of the polynomial (and the root count as a byproduct). It is advisable to explore several system coverings with BEZOUT_GLP before committing to one and calling POLSYS_GLP. A sample main program MAIN_TEMPLATE demonstrates how to use POLSYS_GLP as just described. MAIN_TEMPLATE uses NAMELIST input for the target system and covering definitions, and allows the user to solve multiple polynomial systems in a single run.

The template TARGET_SYSTEM_USER (an external subroutine) is also provided. This subroutine would contain hand-crafted code for function and Jacobian matrix evaluation if the optional argument USER_F_DF to POLSYS_GLP were used.

The system covering must be defined by the user in the module `GLOBAL_GLP`. Heuristics, as in PHCPACK, exist for estimating an optimal system covering (GLP structure), but are no substitute for physical insight into the problem at hand. In practice, polynomial systems typically arise as sums of products with physical variables naturally grouped. Matching the GLP structure to the problem's "physical" structure usually yields a near optimal Bezout number B_{GLP}. Intuitively, the idea is to get all the degrees d_{ij} as low as possible (see the example below). For real problems, an m-homogeneous or PLP partition almost always suffices, and for the remainder a GLP structure is adequate. Of course GPD Bezout numbers can be lower than B_{GLP}, but no class of applications has yet emerged for which B_{GPD} is significantly lower than B_{GLP}.

7.1 INPUT SANITY CHECKING

The target system $F(z) = 0$ is defined by simple tableau input (coefficients and variables' exponents) and the GLP structure is also input; these are stored in the module `GLOBAL_GLP`. Whether the specified GLP structure is consistent with the definition of $F(z)$ is something that must be explicitly verified by the code. For the PLP structure (where the covers P_i are partitions), this consistency check was not hard, since the set degrees d_{ij} for each term in each component of F could be computed, and compared to the purported PLP structure degrees, in one pass of the data. Verifying the GLP structure is nontrivial.

Essentially each monomial $z_1^{\alpha_1} \cdots z_n^{\alpha_n}$ in F_i must have the form $\prod_{j=1}^{m_i} (p_{ij})^{d_{ij}}$, where p_{ij} is a polynomial of degree one in only the variables in the set S_{ij}, or $p_{ij} \equiv 1$. That is, each factor z_k in the monomial must be accounted for by some linear factor p_{ij}. Construct a table with $r = \sum_{j=1}^{n} \alpha_j$ rows labelled by the factors z_k in the monomial, and with $\sum_{j=1}^{m_i} d_{ij}$ columns labelled by the sets S_{ij}, where S_{ij} appears d_{ij} times. A mark in row z_a and column S_{bc} is *legal* if $z_a \in S_{bc}$. The monomial is consistent with the GLP structure if and only if there exists a legal marking of the table with exactly one mark in each row and at most one mark in each column.

The existence of such a table marking can be found by a depth first tree search with backtracking of all the legal table markings $(c_1, c_2, \ldots, c_r)$ in lexicographic order, where c_j is the ordinal index of the column marked in row j.

7.2 AN EXAMPLE

The kinematic synthesis of an RPS serial chain [Su, McCarthy, and Watson 2004] illustrates the value of exploiting a GLP structure by using POLSYS_GLP. The design equations for this chain form a polynomial system of nine quartic equations and one linear equation in 10 unknowns, that is, $F : \mathbf{C}^{10} \to \mathbf{C}^{10}$ given by

$$F(z) = \begin{pmatrix} k_0(\mathbf{P}^2 \cdot \mathbf{P}^2 - \mathbf{P}^1 \cdot \mathbf{P}^1) + 2\mathbf{K} \cdot (\mathbf{P}^2 - \mathbf{P}^1) - (\mathbf{P}^2 \cdot \mathbf{G})^2 + (\mathbf{P}^1 \cdot \mathbf{G})^2 \\ \vdots \\ k_0(\mathbf{P}^{10} \cdot \mathbf{P}^{10} - \mathbf{P}^1 \cdot \mathbf{P}^1) + 2\mathbf{K} \cdot (\mathbf{P}^{10} - \mathbf{P}^1) - (\mathbf{P}^{10} \cdot \mathbf{G})^2 + (\mathbf{P}^1 \cdot \mathbf{G})^2 \\ \mathbf{G} \cdot \mathbf{m} - e \end{pmatrix}.$$

Each $\mathbf{P}^i$ is affine in the unknown parameters (p_1, p_2, p_3), thus $\mathbf{P}_i = [T_i]\mathbf{p}$ for a constant 3×4 matrix $[T_i]$ and $\mathbf{p} = (p_1, p_2, p_3, 1)^t$. Furthermore, the terms $\mathbf{P}^i \cdot \mathbf{P}^i - \mathbf{P}^1 \cdot \mathbf{P}^1$ are affine in (p_1, p_2, p_3) due to cancellation of the quadratic terms, so this term is $[B_i]\mathbf{p}$ for a constant 1×4 matrix $[B_i]$. The vector $\mathbf{G} = (g_1, g_2, g_3)$ provides three unknowns, and $\mathbf{m}$ and e are a known vector and constant, respectively. The remaining unknowns are the vector $\mathbf{K} = (k_1, k_2, k_3)$ and the scalar k_0.

This system can be simplified by eliminating g_3 using the linear equation. The result is

$$\bar{F}(z) = \begin{pmatrix} k_0[B_2]\mathbf{p} + 2\mathbf{K} \cdot [T_2 - T_1]\mathbf{p} - (\mathbf{G} \cdot [T_2]\mathbf{p})^2 + (\mathbf{G} \cdot [T_1]\mathbf{p})^2 \\ \vdots \\ k_0[B_{10}]\mathbf{p} + 2\mathbf{K} \cdot ([T_{10} - T_1]\mathbf{p} - (\mathbf{G} \cdot [T_{10}]\mathbf{p})^2 + (\mathbf{G} \cdot [T_1]\mathbf{p})^2 \end{pmatrix},$$

where the $[T_i]$ and $[B_i]$ are constant matrices. This is a system of nine quartic polynomials in the nine unknowns

$$z = (g_1, g_2, p_1, p_2, p_3, k_0, k_1, k_2, k_3).$$

The 1-homogeneous Bezout number of the system is the total degree of $\bar{F}(z)$ given by $d = 4^9 = 262,144$.

The polynomial system $\bar{F}(z)$ has the covering

$$P = \big\{\{g_1, g_2\}, \{p_1, p_2, p_3\}, \{p_1, p_2, p_3, k_0, k_1, k_2, k_3\}\big\}^9,$$

consisting of the three sets

$$S_{i1} = \{g_1, g_2\},\ S_{i2} = \{p_1, p_2, p_3\},\ S_{i3} = \{p_1, p_2, p_3, k_0, k_1, k_2, k_3\},$$

with the associated degrees

$$d_{i1} = 2,\ d_{i2} = 1,\ d_{i3} = 1; \qquad i = 1, \ldots, 9.$$

This data yields a GLP Bezout number $B_{GLP} = 9,216$ computed by `POLSYS_GLP`. For comparison, the 2-homogeneous Bezout number derived from the partition

$$P = \big\{\{g_1, g_2\}, \{p_1, p_2, p_3, k_0, k_1, k_2, k_3\}\big\}^9$$

with the degree structure

$$d_{i1} = 2,\ d_{i2} = 2; \qquad i = 1, \ldots, 9$$

yields $B_{PLP} = 18,432$, which is twice that of B_{GLP}.

Solving a range of problems with random data using `POLSYS_GLP` reveals 1024 generic (real) roots obtained by tracking $9,216$ homotopy zero curves.

8. PERFORMANCE

Root counting is done in the subroutines `BEZOUT_GLP` and `SINGSYS_GLP`, the former calling the latter. The end game is housed in an internal subroutine `ROOT_GLP`. A careful performance evaluation of the root counting and end game algorithms was done in [Wise, Sommese, and Watson 2000] for POLSYS_PLP, and the performance

of the HOMPACK90 core algorithms is also well documented. Perhaps of most interest is the parallel performance of POLSYS_GLP, so some large scale parallel results are given here. For the example in Section 7.2, POLSYS_GLP has been tested on several parallel architectures, namely, the IBM Power4 computer (called DataStar) located at the San Diego Supercomputing Center, a Linux cluster of computers (called MPC) with AMD Opteron processors located at the University of California at Irvine, and the IBM Power3 computer (denoted MSI_SP) at the Minnesota Supercomputing Institute. Table I shows the maximum execution times in seconds (first number in columns 2–4) and speedups (second number in columns 2–4) on these parallel computing platforms for 4, 8, 16, 24, and 32 processors. Note that, for consistency across all the computers, the speedups are computed relative to a four processor execution time, since on the MSI_SP, any run on fewer than four processors took more than six hours to complete and did not fit the site's queue policy. Also, due to access constraints on the MPC, a 32 processor run was not possible on this system, which accounts for the asterisk in the last row of the MPC column.

Table I. Parallel Performance of POLSYS_GLP on Three Different Architectures

Proc.	DataStar	MPC	MSI_SP
4	4.253E+03, 1.00	3.872E+03, 1.00	1.564E+04, 1.00
8	1.826E+03, 2.33	1.661E+03, 2.33	6.714E+03, 2.33
16	8.531E+02, 4.98	7.765E+02, 4.99	3.134E+03, 4.99
24	5.552E+02, 7.66	4.963E+02, 7.80	2.055E+03, 7.61
32	4.130E+02, 10.30	*	1.521E+03, 10.28

The speedups are exactly what one would expect with a master/slave paradigm where the master processor is not also a slave. For instance, in going from four to eight processors, the number of slaves goes from three to seven, and 7/3 is exactly the speedup observed. Timings on MPC are problematic, because of varying switch configurations for different numbers of processors. This accounts for the inconsistent speedups for MPC in Table I. The master overhead is negligible, since the speedup is nearly perfect through 32 processors, indicating consistently good performance of POLSYS_GPL on large scale problems and parallel architectures of varying power. It is noticeable, for example, that DataStar and MPC execution times are an order of magnitude better than those on MSI_SP. This may be explained by the difference in processor speeds—for DataStar and MPC, they are 1.7 GHz and 1.4 GHz, respectively, whereas the MSI_SP Power3 processors run at only 222 MHz (on NightHawk nodes). The amount of memory was not a limiting factor for the test problem—all the computers had about the same amount of memory available to POLSYS_GLP (1 GB/processor). Communications were performed over high speed interconnection networks. In particular, communications utilize the proprietary IBM Switch2 across the 4-way SMP nodes of the MSI_SP and the IBM Federation switch across the 8-way SMP nodes of the DataStar, while

shared memory is used within each node. For the MPC system, Gigabit Ethernet is used across dual processor nodes along with shared memory for intranode communications. It was observed that the master processor did not present a bottleneck for these experiments, since none of the processors incurred significant idle time.

ACKNOWLEDGEMENT

The authors are deeply indebted to A. P. Morgan, A. J. Sommese, and C. W. Wampler for years of collaboration upon which this work is based.

BIBLIOGRAPHY

Allgower, E. L., and Georg, K. 1990. *Numerical Continuation Methods*. Springer-Verlag, Berlin.

Allison, D. C. S., Harimoto, S., and Watson, L.T. 1989. The granularity of parallel homotopy algorithms for polynomial systems of equations. *Internat. J. Comput. Math. 29*, 21–37.

Allison, D. C. S., Chakraborty, A., and Watson, L. T. 1989. Granularity issues for solving polynomial systems via globally convergent algorithms on a hypercube. *J. Supercomputing, 3*, 5–20.

Anderson, E., Bai, Z., Bishof, C., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., Ostrouchov, S., and Sorensen, D. 1995. *LAPACK User's Guide*. SIAM, Philadelphia, PA, 2nd ed.

Björk, G., and Fröberg, R. 1991. A faster way to count the solutions of inhomogeneous systems of algebraic equations, with applications to cyclic n-roots. *J. Symbolic Computation 12*, 329–336.

Blum, L., Cucker, F., Shub, M., and Smale, S. 1998. *Complexity and Real Computation*. Springer-Verlag, New York.

Chakraborty, A., Allison, D. C. S., Ribbens, C. J., and Watson, L. T. 1991. Note on unit tangent vector computation for homotopy curve tracking on a hypercube. *Parallel Comput. 17*, 1385–1395.

Chakraborty, A., Allison, D. C. S., Ribbens, C. J., and Watson, L. T. 1993. The parallel complexity of embedding algorithms for the solution of systems of nonlinear equations. *IEEE Trans. Parallel Distrib. Systems 4*, 458–465.

Chow, S. N., Mallet-Paret, J., and Yorke, J. A. 1979. A homotopy method for locating all zeros of a system of polynomials. In *Functional Differential Equations and Approximation of Fixed Points*, Lecture Notes in Math. #730, H. O. Peitgen and H. O. Walther, Eds., Springer-Verlag, 228–237.

Drexler, F. J. 1979. Eine methode zur berechung sämtlicher lösunger von polynomgleichungessystemen. *Numer. Math. 29*, 45–58.

Garcia, C. B., and Zangwill, W. I. 1977. Global continuation methods for finding all solutions to polynomial systems of equations in N variables. Center for Math Studies in Business and Economics Report No. 7755, Univ. of Chicago, Chicago, IL.

Griewank, A. 1985. On solving nonlinear equations with simple singularities or nearly singular solutions. *SIAM Rev. 27*, 537–563.

Lundberg, B. N., and Poore, A. B. 1991. Variable order Adams-Bashforth predictors with an error-stepsize control for continuation methods. *SIAM J. Sci. Stat. Comput. 12*, 695–723.

Meintjes, K., and Morgan, A. P. 1985. A methodology for solving chemical equilibrium systems. Tech. Rep. GMR-4971, GM Res. Lab., Warren, MI.

Morgan, A. P. 1983. A method for computing all solutions to systems of polynomial equations. *ACM. Trans. Math. Software 9*, 1–17.

Morgan, A. P. 1986a. A transformation to avoid solutions at infinity for polynomial systems. *Appl. Math. Comput. 18*, 77–86.

MORGAN, A. P. 1986b. A homotopy for solving polynomial systems. *Appl. Math. Comput. 18*, 87–92.

MORGAN, A. P. 1987. *Solving polynomial systems using continuation for engineering and scientific problems*. Prentice-Hall, Englewood Cliffs, NJ.

MORGAN, A. P., AND SOMMESE, A. J. 1987a. A homotopy for solving general polynomial systems that respects m-homogeneous structures. *Appl. Math. Comput. 24*, 101–113.

MORGAN, A. P., AND SOMMESE, A. J. 1987b. Computing all solutions to polynomial systems using homotopy continuation. *Appl. Math. Comput. 24*, 115–138.

MORGAN, A. P., SOMMESE, A. J., AND WAMPLER, C. W. 1991. Computing singular solutions to nonlinear analytic systems. *Numer. Math. 58*, 669–684.

MORGAN, A. P., SOMMESE, A. J., AND WAMPLER, C. W. 1992a. Computing singular solutions to polynomial systems. *Advances Appl. Math. 13*, 305–327.

MORGAN, A. P., SOMMESE, A. J., AND WAMPLER, C. W. 1992b. A power series method for computing singular solutions to nonlinear analytic systems. *Numer. Math. 63*, 391–409.

MORGAN, A. P., SOMMESE, A. J., AND WAMPLER, C. W. 1995. A product-decomposition bound for Bezout numbers. *SIAM J. Numer. Anal. 32*, 1308–1325.

MORGAN, A. P., SOMMESE, A. J., AND WATSON, L. T. 1989. Finding all isolated solutions to polynomial systems using HOMPACK. *ACM Trans. Math. Software 15*, 93–122.

MORGAN, A. P., AND WATSON, L. T. 1989. A globally convergent parallel algorithm for zeros of polynomial systems. *Nonlinear Anal. 13*, 1339–1350.

PELZ, W. AND WATSON, L. T. 1989. Message length effects for solving polynomial systems on a hypercube. *Parallel Comput. 10*, 161–176.

SOSONKINA, M., WATSON, L. T., AND STEWART, D. E. 1996. Note on the end game in homotopy zero curve tracking. *ACM Trans. Math. Software 22*, 281–287.

SU, H.-J., MCCARTHY, J. M., AND WATSON, L. T. 2004. Generalized linear product homotopy algorithms and the computation of reachable surfaces. *ASME J. Comput. Information Sci. Engrg.*, in press.

TSAI, L.-W., AND MORGAN, A. P. 1985. Solving the kinematics of the most general six- and five-degree-of-freedom manipulators by continuation methods. *ASME J. Mechanisms, Transmissions, Aut. Design 107*, 48–57.

VAN DER WAERDEN, B. L. 1953. *Modern Algebra*. Vols. 1, 2, Frederick Ungar Pub. Co., New York.

VERSCHELDE, J. May, 1996. Homotopy continuation methods for solving polynomial systems. Ph.D. Thesis, Dept. of Computer Sci., Katholieke Univ. Lueven, Lueven, Belgium.

VERSCHELDE, J. 1997. PHCPACK: A general-purpose solver for polynomial systems by homotopy continuation. Preprint.

VERSCHELDE, J., AND COOLS, R. 1993. Symbolic homotopy construction. *Appl. Algebra Engrg. Comm. Comput. 4*, 169–183.

VERSCHELDE, J., AND HAEGEMANS, A. 1993. The GBQ-algorithm for constructing start systems of homotopies for polynomial systems. *SIAM J. Numer. Anal. 30*, 583–594.

WAMPLER, C. W. 1994. An efficient start system for multi-homogeneous polynomial continuation. *Numer. Math. 66*, 517–523.

WATSON, L. T., BILLUPS, S. C., AND MORGAN, A. P. 1987. Algorithm 652: HOMPACK: A suite of codes for globally convergent homotopy algorithms. *ACM Trans. Math. Software 13*, 281–310.

WATSON, L. T., SOSONKINA, M., MELVILLE, R. C., MORGAN, A. P., AND WALKER, H. F. 1997. Algorithm 777: HOMPACK90: A suite of Fortran 90 codes for globally convergent homotopy algorithms. *ACM Trans. Math. Software 23*, 514–549.

WISE, S. M., SOMMESE, A. J., AND WATSON, L. T. 2000. Algorithm 801: POLSYS_PLP: A partitioned linear product homotopy code for solving polynomial systems of equations. *ACM Trans. Math. Software 26*, 176–200.

WRIGHT, A. H. 1985. Finding all solutions to a system of polynomial equations. *Math. Comp. 44*, 125–133.